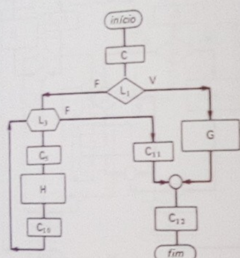
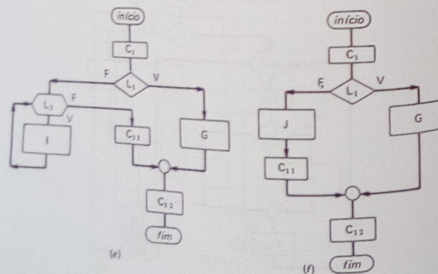
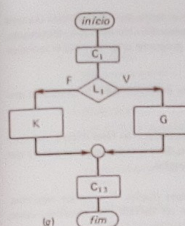
Na passagem da original para (*a*), substituímos as seqüências de comandos "C_8; C_1", "C_6; C_7" e "C_3; C_4" pelas "caixas" *A*, *B* e *C*, respectivamente. Na passagem por (*b*), reconhecemos duas estruturas enquanto-faça que novamente podemos substituir pelas duas "caixas" *D* e *E*. Continuamos assim até que na figura (*i*) o fluxograma consiste simplesmente de uma caixa-preta, cuja função é a função total do algoritmo.



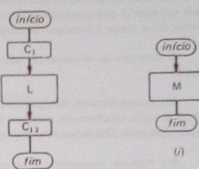
(*d*)



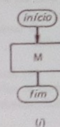
(*e*) (*f*)



(*g*)



(*h*)



(*i*)

Com um algoritmo não estruturado não teria sido possível fazer esta decomposição. Temos então um método para testar se um dado fluxograma é estruturado ou não.

Se consideramos a série da ordem inversa, de (*i*) até o original, ela corresponde extamente à construção do algoritmo por refinamento sucessivo. Inicialmente, o algoritmo consiste numa caixa-preta *M*, cuja função é definida pela especificação do algoritmo. Das várias possibilidades de se estruturar essa caixa-preta, verifica-se que uma seqüência de três subfunções C_1, L e C_{12} é adequada (ou seja, que o processo de solução consiste na execução em seqüência temporal de três subprocessos). Enquanto que C_1 e C_{12} já são comandos básicos que não precisam mais ser refinados (por exemplo, comandos de leitura e de impressão), a caixa *L* ainda tem que ser refinada, no nosso caso, mediante uma alternativa. Dependendo da condição L_1 ser verdadeira ou não, a execução de *G* ou de *K*, respectivamente, vai resolver o problema. Esse processo é continuado até que todas as "caixas" representem comandos básicos.

A chave da produção de programas confiáveis está na adoção de uma atitude humilde por parte do programador: "reconhecer que tem uma cabeça muito pequena e que é melhor tratar de conviver com ela e respeitar suas limitações" (Dijkstra). Há quem diga que o limite máximo de tamanho dos programas compreensíveis é de ordem de uma página de listagem (é muito importante poder examinar os programas sem ter que efetuar perturbador manuseio de folhas).

Ora, a maioria dos programas de utilidade prática tem, quando codificado numa das linguagens disponíveis, extensão muito maior que uma página. Se não tivessem, seriam tão simples que não se precisaria de programação estruturada nenhuma para desenvolvê-los. Como são extensos, a solução é aplicar o que Dijkstra chama de "regra áurea": *dividir para reinar*, ou seja, desenvolve-se o programa numa série de etapas chamadas *refinamentos*.

No primeiro refinamento, o programa é repartido numa seqüência de trechos, onde cada trecho descreve uma ação mais elementar que a do programa; a concatenação das ações dos trechos deve produzir a ação total desejada do programa.

A ação de cada trecho pode eventualmente ser suficientemente simples para que se possa exprimi-la diretamente com um pequeno texto na pseudolinguagem. Se isto acontecer, muito